Philipp CYO NBA

Nico Philipp

2/12/2024

NBA Playoffs Project

Introduction and Project Aim

The aim of this project is to predict weather or not an nba team made the playoffs or not over a 6 season span, based on publically available basketball statistics.

Publicly available basketball statistics were collected using the nbastatR package for the NBA seasons 2017, 2018, 2019, 2021, 2022, and 2023. The data includes various performance metrics for each NBA team. The data was cleaned, selecting relevant variables such as assists per game, blocks per game, defensive rebounds per game, offensive rebounds per game, 2-point and 3-point field goal percentages, free-throw percentage, points scored per game, year of the season, and the name of the NBA team.

Machine learning-based decision trees (classification) were used to analyze and visualize the data. While a random forrest approach might provide us with more accurate or stable models, they are often harder to interpret for sport practitioners such as coaches and scouts. Therefore, given their interpretability, decision trees were chosen for this project. Furhter, a Receiver Operators Curve (ROC) statistic was calculated and visualized highlighting the models ability to distinguish between non-playoff and playoff teams. To fulfill the two-model requirement for this assignment, I will also adopt a more straight forward logistic regression approach, supplemented with an ROC analysis.

Dependent/Response Variable:

- Playoffs (True vs. FALSE)

Independent/Predictor Variables:

- astPerGameTeam (Assists per game)
- blkPerGameTeam (Blocks per game)
- drbPerGameTeam (Defensive rebounds per game)
- orbPerGameTeam (Offensive rebounds per game)
- pctFG2PerGameTeam (2-point field goal percentage)
- pctFG3PerGameTeam (3-point field goal percentage)
- pctFTPerGameTeam (Free-throw percentage)
- ptsPerGameTeam (Points scored per game)

Other variables:

- yearSeason (Year of season)
- nameTeam (Name of nba team)

```r
#load libraries
library(tidyverse)
library(nbastatR)
library(tidymodels)
library(rpart)
library(rpart.plot)
library(ROCR)
library(caret)
library(tinytex)
library(pROC)

Sys.setenv(VROOM_CONNECTION_SIZE = 500000)# Set the connection buffer size
```

```
suppressMessages({
#gather data from the nbastatr package
  nbastatR::bref_teams_stats(seasons = 2017,
                               return_message = FALSE,
                               assign_to_environment = TRUE,
                               nest_data = FALSE,
                               join_data = TRUE,
                               widen_data = TRUE)
dat1 <- dataBREFPerGameTeams

  nbastatR::bref_teams_stats(seasons = 2018,
                               return_message = FALSE,
                               assign_to_environment = TRUE,
                               nest_data = FALSE,
                               join_data = TRUE,
                               widen_data = TRUE)
dat2 <- dataBREFPerGameTeams

  nbastatR::bref_teams_stats(seasons = 2019,
                               return_message = FALSE,
                               assign_to_environment = TRUE,
                               nest_data = FALSE,
                               join_data = TRUE,
                               widen_data = TRUE)
dat3 <- dataBREFPerGameTeams

  nbastatR::bref_teams_stats(seasons = 2021,
                               return_message = FALSE,
                               assign_to_environment = TRUE,
                               nest_data = FALSE,
                               join_data = TRUE,
                               widen_data = TRUE)
dat4 <- dataBREFPerGameTeams

  nbastatR::bref_teams_stats(seasons = 2022,
                               return_message = FALSE,
                               assign_to_environment = TRUE,
                               nest_data = FALSE,
                               join_data = TRUE,
                               widen_data = TRUE)
dat5 <- dataBREFPerGameTeams

nbastatR::bref_teams_stats(seasons = 2023,
                               return_message = FALSE,
                               assign_to_environment = TRUE,
                              nest_data = FALSE,
                              join_data = TRUE,
                              widen_data = TRUE)
dat6 <- dataBREFPerGameTeams

#combine into one dataframe
combined_df <- rbind(dat1, dat2, dat3, dat4, dat5, dat6)
})
```

```
## StandingsConf
## Assigning NBA player dictionary to df_dict_nba_players to your environment
## StandingsDiv
## PerGame
## Totals
## PerPoss
## Shooting
## StandingsConf
## StandingsDiv
## PerGame
## Totals
## PerPoss
## Shooting
## StandingsConf
## StandingsDiv
## PerGame
## Totals
## PerPoss
## Shooting
## StandingsConf
## StandingsDiv
## PerGame
## Totals
## PerPoss
## Shooting
## StandingsConf
## StandingsDiv
## PerGame
## Totals
## PerPoss
## Shooting
## StandingsConf
## StandingsDiv
## PerGame
## Totals
## PerPoss
## Shooting
```

```
#select metrics of interest
suppressMessages({
combined_df_clean <- combined_df %>%
  select(yearSeason, nameTeam, isPlayoffTeam, astPerGameTeam, blkPerGameTeam, drbPerGameTeam, orbPerGam

})
```

```
#playoff vs. non playoff counts
playoffs_counts <- table(combined_df_clean$isPlayoffTeam)
print(playoffs_counts)
```

```
##
## FALSE  TRUE
##    80   100
```

In our data, 80 teams missed the playoffs, while 100 teams made the playoffs over the span of 6 seasons.

Analysis 1: Decision Tree Model

```
#start decision tree model process

#80 percent train, 20 percent test
set.seed(123)

trainIndex <- createDataPartition(combined_df_clean$isPlayoffTeam, p = 0.8, list = FALSE, times = 1)

train_df <- combined_df_clean[trainIndex,]
test_df <- combined_df_clean[-trainIndex,]

#Defining our predictor (x) and response (y) variables
x <- combined_df_clean %>% select(c(yearSeason, nameTeam, astPerGameTeam, blkPerGameTeam, drbPerGameTea

y <- combined_df_clean$isPlayoffTeam

#Extracting Training Data (X_train, y_train) and Validation Data (X_val, y_val)

X_train <- train_df %>% select(c(yearSeason, nameTeam, astPerGameTeam, blkPerGameTeam, drbPerGameTeam, 

y_train <- train_df$isPlayoffTeam

###

X_val <- test_df %>% select(c(yearSeason, nameTeam, astPerGameTeam, blkPerGameTeam, drbPerGameTeam, orb

y_val <- test_df$isPlayoffTeam
```

A 10-fold cross-validation with leave-one-out strategy is utilized to assess the model's accuracy and generalization performance

---

```
#define the parameters for model training.
set.seed(123)

fitControl <- trainControl(method = 'loocv',
                           number = 10,
                           savePredictions = 'final') #Save final prediction for each fold

#
view(train_df)

#Training the decision tree model
#set the max depth of the tree, and minimum instances for the tree to split
set.seed(123)

tree <- train(factor(isPlayoffTeam) ~.,
              data = train_df %>% select(-c("nameTeam", "yearSeason")),
              method = 'rpart',
              metric = 'Accuracy',
              trControl = fitControl,
              control = rpart.control(minsplit = 1,
                                      maxdepth = 4))

tree
```

```
## CART
##
## 144 samples
##   8 predictor
##   2 classes: 'FALSE', 'TRUE'
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 143, 143, 143, 143, 143, 143, ...
## Resampling results across tuning parameters:
##
##   cp        Accuracy   Kappa
##   0.046875  0.7222222  0
##   0.109375  0.6875000  0
##   0.390625  0.4652778  0
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.046875.
```

```
#This code segment is evaluating the performance of a decision tree model on different datasets (traini

#train - test - split

#determining if overfit(accuracy < train ) or underfit(accurary>train)

#1) setting up model for the trained dataframe
tree_predict_train <- tree %>% predict(X_train, method = 'prop')
#2) testing model on the validating dataframe - train/test split.
tree_predict_val <- tree %>% predict(X_val, method = 'prop')
#3) testing model on the whole dataframeix
tree_predict <- tree %>% predict(x, method = 'prop')

###

#1) testing model on the trained dataframe - theoretically the best fit bc no new data. Over/underfit i
```

```r
confusionMatrix(factor(tree_predict_train), factor(y_train), positive = "TRUE")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    48   12
##      TRUE     16   68
##
##                Accuracy : 0.8056
##                  95% CI : (0.7314, 0.8667)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : 2.558e-10
##
##                   Kappa : 0.6038
##
##  Mcnemar's Test P-Value : 0.5708
##
##             Sensitivity : 0.8500
##             Specificity : 0.7500
##          Pos Pred Value : 0.8095
##          Neg Pred Value : 0.8000
##              Prevalence : 0.5556
##          Detection Rate : 0.4722
##    Detection Prevalence : 0.5833
##       Balanced Accuracy : 0.8000
##
##        'Positive' Class : TRUE
##
```

```r
#2) testing model on the validating dataframe - train/test split.
confusionMatrix(factor(tree_predict_val), factor(y_val))
```

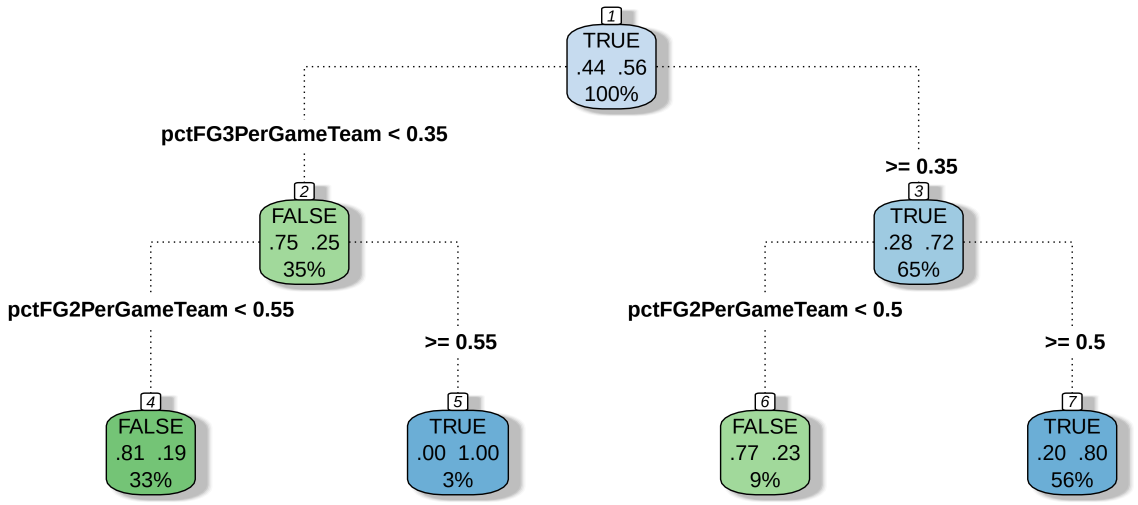
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    12    4
##      TRUE      4   16
##
##                Accuracy : 0.7778
##                  95% CI : (0.6085, 0.8988)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : 0.004838
##
##                   Kappa : 0.55
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.7500
##             Specificity : 0.8000
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.8000
##              Prevalence : 0.4444
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.4444
##       Balanced Accuracy : 0.7750
##
##        'Positive' Class : FALSE
##
```

```
#3) testing model on the whole dataframe - report the model's score metrics from this confusionmatrix.
confusionMatrix(factor(tree_predict), factor(y), positive = "TRUE")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    60   16
##      TRUE     20   84
##
##                Accuracy : 0.8
##                  95% CI : (0.734, 0.8558)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : 5.114e-12
##
##                   Kappa : 0.593
##
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.8400
##             Specificity : 0.7500
##          Pos Pred Value : 0.8077
##          Neg Pred Value : 0.7895
##              Prevalence : 0.5556
##          Detection Rate : 0.4667
##    Detection Prevalence : 0.5778
##       Balanced Accuracy : 0.7950
##
##        'Positive' Class : TRUE
##
```

The decision tree model achieved an accuracy of approximately 80% in predicting playoff outcomes. The confusion matrices for the training, validation, and entire datasets provide a detailed breakdown of the model's performance.

```
#plot decision tree
rattle::fancyRpartPlot(tree$finalModel, type = 4, caption = "Playoff Model")
```



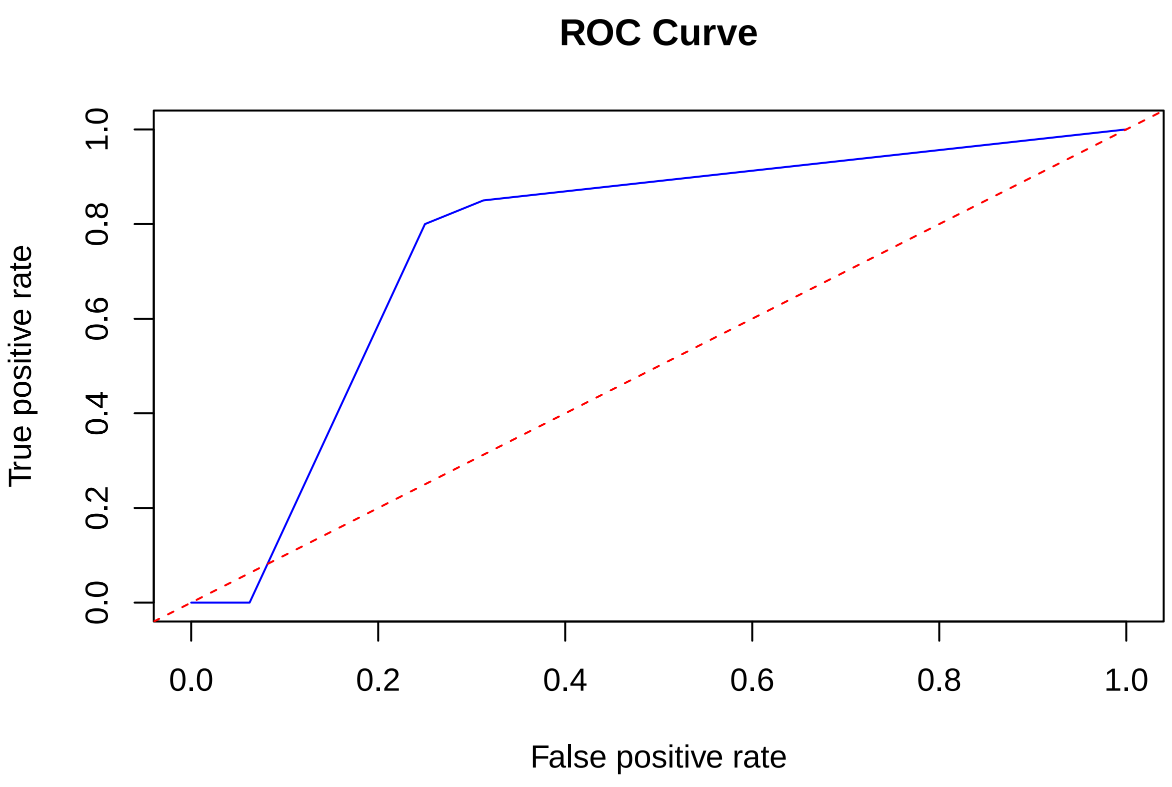
Playoff Model

The decision tree plot (Figure 1) illustrates the features of importance for predicting playoff participation. Notably, teams with a 3-point field goal percentage of 35% or higher had a 72% chance of making the playoffs. Further, teams with a 3-point percentage of 35% or higher and a two-point percentage of 50% or higher had an 80% chance of making the playoffs.

```
#Receiver Operating Characteristic (ROC) curve analysis to evaluate the performance of a binary classif

# Get predicted probabilities for the positive class (isPlayoffTeam = TRUE)
set.seed(123)

predictions <- predict(tree, newdata = X_val, type = "prob")[, "TRUE"]

#create prediction object
pred <- prediction(predictions, y_val)

# Compute the AUC
auc <- performance(pred, "auc")@y.values[[1]]

# Print the AUC
print(auc)
```

```
## [1] 0.7625
```

```
#calculate performance measures
perf <- performance(pred, "tpr", "fpr")

#plot roc curve
plot(perf, main = "ROC Curve", col = "blue")

# Add diagonal reference line
abline(a = 0, b = 1, col = "red", lty = 2)
```



The Receiver Operating Characteristic (ROC) curve analysis provides insights into the model's ability to discriminate between playoff and non-playoff teams. The Area Under the Curve (AUC) is a key metric indicating the model's discriminatory power:

To fullfil the two model requirement, we want to see how a more simple, logistic regression model compares to the decision tree model.

Analysis 2: Logistic Regression

```r
# Logistic Regression Model
set.seed(123)
logistic_model <- glm(isPlayoffTeam ~ ., data = train_df, family = "binomial")

# Predictions on Training Set
logistic_predict_train <- predict(logistic_model, newdata = X_train, type = "response")

# Predictions on Validation Set
logistic_predict_val <- predict(logistic_model, newdata = X_val, type = "response")

# Predictions on the Whole Dataset
logistic_predict <- predict(logistic_model, newdata = x, type = "response")

# Model Evaluation for Logistic Regression
confusionMatrix(factor(logistic_predict_train > 0.5), factor(y_train), positive = "TRUE")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    57    7
##      TRUE      7   73
##
##                Accuracy : 0.9028
##                  95% CI : (0.8423, 0.9458)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8031
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9125
##             Specificity : 0.8906
##          Pos Pred Value : 0.9125
##          Neg Pred Value : 0.8906
##              Prevalence : 0.5556
##          Detection Rate : 0.5069
##    Detection Prevalence : 0.5556
##       Balanced Accuracy : 0.9016
##
##        'Positive' Class : TRUE
##
```

```
confusionMatrix(factor(logistic_predict_val > 0.5), factor(y_val))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    11    6
##      TRUE      5   14
##
##                Accuracy : 0.6944
##                  95% CI : (0.5189, 0.8365)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : 0.06397
##
##                   Kappa : 0.3851
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.6875
##             Specificity : 0.7000
##          Pos Pred Value : 0.6471
##          Neg Pred Value : 0.7368
##              Prevalence : 0.4444
##          Detection Rate : 0.3056
##    Detection Prevalence : 0.4722
##       Balanced Accuracy : 0.6937
##
```

```
##        'Positive' Class : FALSE
##
```

```
confusionMatrix(factor(logistic_predict > 0.5), factor(y), positive = "TRUE")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    68   13
##      TRUE     12   87
##
##                Accuracy : 0.8611
##                  95% CI : (0.8018, 0.9081)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7191
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8700
##             Specificity : 0.8500
##          Pos Pred Value : 0.8788
##          Neg Pred Value : 0.8395
##              Prevalence : 0.5556
##          Detection Rate : 0.4833
##    Detection Prevalence : 0.5500
##       Balanced Accuracy : 0.8600
##
##        'Positive' Class : TRUE
##
```

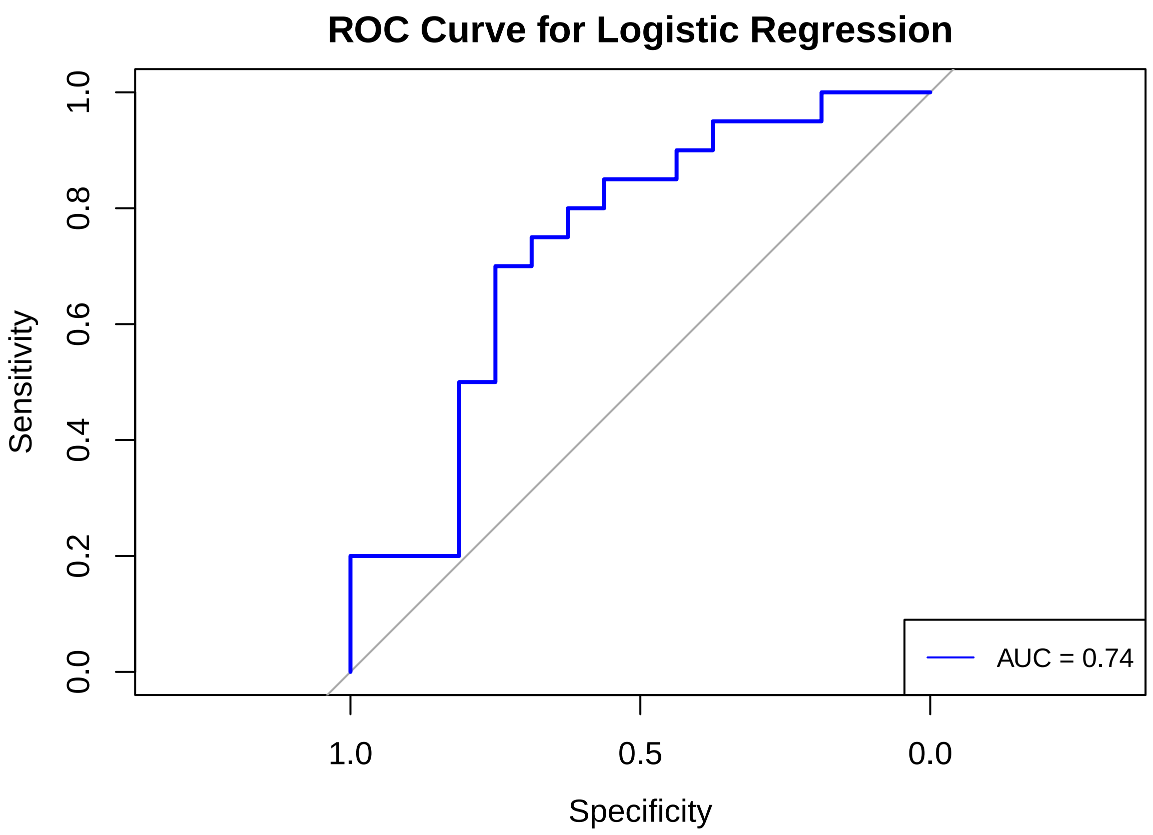
Plot the ROC results from the logistic regression model

```
# ROC Curve for Logistic Regression
roc_curve <- roc(y_val, logistic_predict_val)
```

```
## Setting levels: control = FALSE, case = TRUE
```

```
## Setting direction: controls < cases
```

```
auc <- auc(roc_curve)

# Plotting ROC Curve
plot(roc_curve, main = "ROC Curve for Logistic Regression", col = "blue")

legend("bottomright", legend = paste("AUC =", round(auc, 2)), col = "blue", lty = 1, cex = 0.8)
```



Summary Paragraph:

The decision tree model presented in this project showed that it was able to predict weather a team would make the playoffs with an accuracy of about 80%. More interestingly to practitioners may be the features of importance identified within the decision tree figure. More specifically, the figure shows that if a team shoots three pointers with a percentage of 35 or more percent, in 72% of cases they will reach the playoffs. If we go one step further, and teams that shoot 35 or more percent from three also shoots two point field goals with 50% or higher, the chances of reaching the playoffs goes up to 80%. This is in agreement with previous peer-reviewed literature, comparing winning to losing teams in the NBA (Cabarkapa et al. 2022). On the other hand, only shooting 35% or below from three leads to an 75% chance of not making the playoffs based on our model.

Interestingly, our model has a higher sensitivity, compared to specificity, meaning it is better at ruling in playoff participation, rather than ruling out playoff participation. This seems positive, given that coaches tend to be more interested in key performance indicators predicting making the playoffs vs. missing the playoffs. The data presented in this project may be of acute interest to coaches and scouts, giving actionable insights into basketball statistics distinguishing between playoff and non-playoff teams at the nba level of play.

Compared to the slightly more complex decision tree model, the logistic regression model yields a higher accuracy (86%). However, the accuracy on the validation data set is lower (69%), compared the validation data set in the decision tree model (78%), suggesting slight overfitting. Area under the curve (AUC) values are comparable between the two approaches. However, the decision tree model presents with a nice visual, giving actionable insights into the findings to respective stakeholders (e.g., coaches and scouts).

References

Cabarkapa D, Deane MA, Fry AC, Jones GT, Cabarkapa DV, Philipp NM, et al. Game statistics that discriminate winning and losing at the NBA level of basketball competition. PLoS One. 2022 Aug 19;17(8):e0273427.

Ayasdi. (2018). nbastatR: Advanced NBA Statistics. R package version 0.0.3. https://github.com/ayasdi/nb

---

While the primary analysis for this project included logistic regression and decision tree modelling, I am also interested in applying a random forrest regression approach

Let's lastly try a random forrest regression model to see if we can improve findings

---

Analysis 3: Random Forrest Regression

```r
#Load required libraries

library(randomForest)
library(caret)
library(ranger)
```

Start Modelling Process

```r
set.seed(123)

# Train-Test Split
trainIndex <- createDataPartition(combined_df_clean$isPlayoffTeam, p = 0.8, list = FALSE, times = 1)

train_df <- combined_df_clean[trainIndex,]
test_df <- combined_df_clean[-trainIndex,]

# Define predictor (x) and response (y) variables
x <- combined_df_clean %>% select(c(yearSeason, nameTeam, astPerGameTeam, blkPerGameTeam, drbPerGameTea
y <- combined_df_clean$isPlayoffTeam

# Extract Training Data (X_train, y_train) and Validation Data (X_val, y_val)
X_train <- train_df %>% select(c(yearSeason, nameTeam, astPerGameTeam, blkPerGameTeam, drbPerGameTeam, 
y_train <- train_df$isPlayoffTeam

X_val <- test_df %>% select(c(yearSeason, nameTeam, astPerGameTeam, blkPerGameTeam, drbPerGameTeam, orb
y_val <- test_df$isPlayoffTeam
```

Define model parameters

```r
# Define parameters for model training
set.seed(123)

fitControl <- trainControl(method = 'cv', number = 10, savePredictions = 'final') # Save final predicti

# Define a grid of hyperparameters to search over
grid <- expand.grid(mtry = c(2, 3, 4), # Try different values of mtry
                    splitrule = c("gini", "extratrees"),
                    min.node.size = c(1, 5, 10))
```

```r
# Training the Random Forest model
rf_model <- train(factor(isPlayoffTeam) ~ .,
                  data = train_df %>% select(-c("nameTeam", "yearSeason")),
                  method = 'ranger',
                  metric = 'Accuracy',
                  trControl = fitControl,
                  importance = 'impurity',
                  tuneGrid = grid)

# Evaluate the performance of the Random Forest model on different datasets
rf_predict_train <- predict(rf_model, newdata = X_train)
rf_predict_val <- predict(rf_model, newdata = X_val)
rf_predict <- predict(rf_model, newdata = x)

# Generate confusion matrices
confusionMatrix(factor(rf_predict_train), factor(y_train), positive = "TRUE")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    56    6
##      TRUE      8   74
##
##                Accuracy : 0.9028
##                  95% CI : (0.8423, 0.9458)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8025
##
##  Mcnemar's Test P-Value : 0.7893
##
##             Sensitivity : 0.9250
##             Specificity : 0.8750
##          Pos Pred Value : 0.9024
##          Neg Pred Value : 0.9032
##              Prevalence : 0.5556
##          Detection Rate : 0.5139
##    Detection Prevalence : 0.5694
##       Balanced Accuracy : 0.9000
##
##        'Positive' Class : TRUE
##
```

```r
confusionMatrix(factor(rf_predict_val), factor(y_val))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    13    4
##      TRUE      3   16
##
```

```
##                Accuracy : 0.8056
##                  95% CI : (0.6398, 0.9181)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : 0.001559
##
##                   Kappa : 0.6087
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.8125
##             Specificity : 0.8000
##          Pos Pred Value : 0.7647
##          Neg Pred Value : 0.8421
##              Prevalence : 0.4444
##          Detection Rate : 0.3611
##    Detection Prevalence : 0.4722
##       Balanced Accuracy : 0.8063
##
##        'Positive' Class : FALSE
##
```

```
confusionMatrix(factor(rf_predict), factor(y), positive = "TRUE")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    69   10
##      TRUE     11   90
##
##                Accuracy : 0.8833
##                  95% CI : (0.8272, 0.9263)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7635
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9000
##             Specificity : 0.8625
##          Pos Pred Value : 0.8911
##          Neg Pred Value : 0.8734
##              Prevalence : 0.5556
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.5611
##       Balanced Accuracy : 0.8813
##
##        'Positive' Class : TRUE
##
```

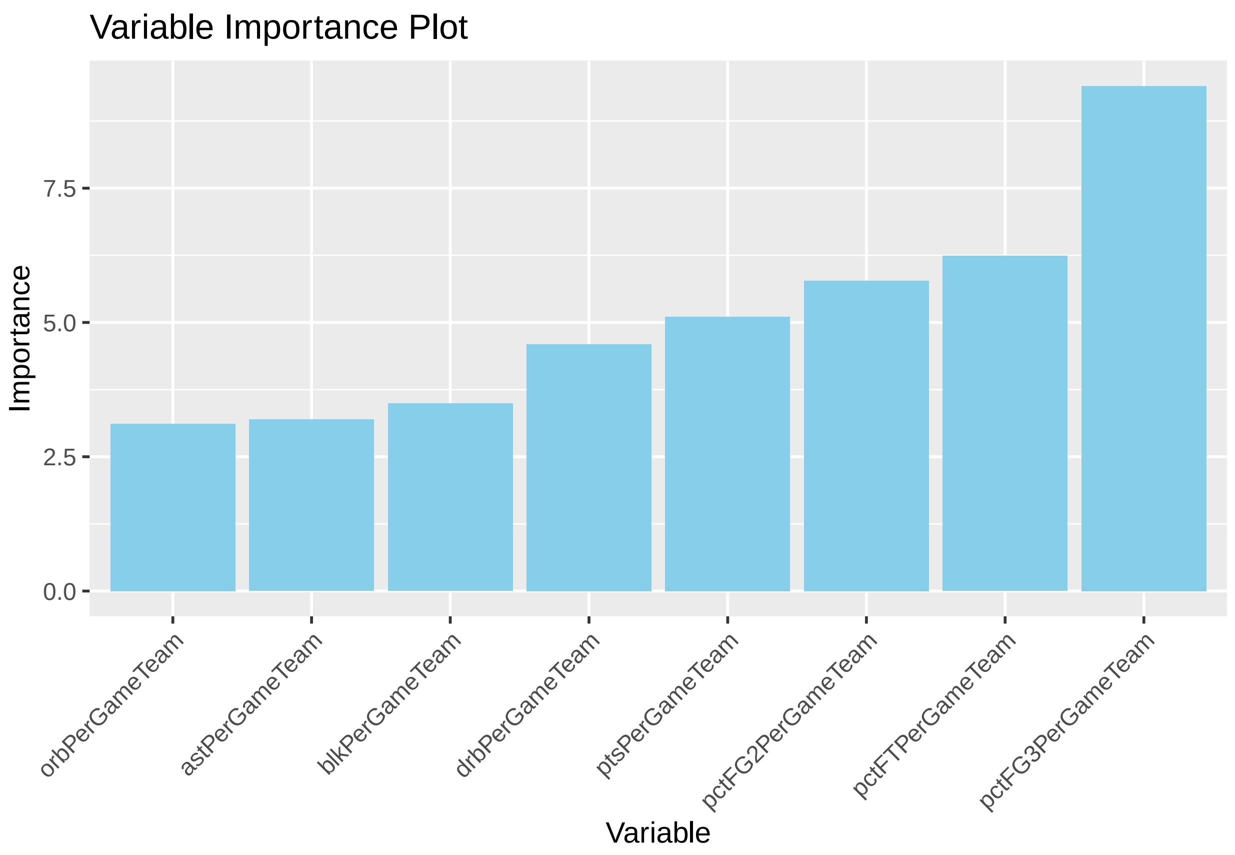
The above findings suggest that the random forrest approach leads to the best predictive model.

Identify variable importance

```
###variable importance

# Plot Variable Importance
var_importance <- rf_model$finalModel$variable.importance

# Convert variable.importance to a data frame
var_importance_df <- data.frame(
  Variable = names(var_importance),
  Importance = var_importance
)

# Order the data frame by Importance in decreasing order
var_importance_df <- var_importance_df[order(-var_importance_df$Importance), ]

# Print the variable importance data frame
print(var_importance_df)
```

```
##                            Variable Importance
## pctFG3PerGameTeam pctFG3PerGameTeam   9.398687
## pctFTPerGameTeam   pctFTPerGameTeam   6.239452
## pctFG2PerGameTeam pctFG2PerGameTeam   5.777074
## ptsPerGameTeam       ptsPerGameTeam   5.109042
## drbPerGameTeam       drbPerGameTeam   4.594454
## blkPerGameTeam       blkPerGameTeam   3.491372
## astPerGameTeam       astPerGameTeam   3.194537
## orbPerGameTeam       orbPerGameTeam   3.111199
```

```
# Plot the variable importance
ggplot(var_importance_df, aes(x = reorder(Variable, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(x = "Variable", y = "Importance", title = "Variable Importance Plot") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Compare All 3 Models

```r
# Create a table for model evaluation results
accuracy_scores <- data.frame(
  Model = c("Decision Tree", "Logistic Regression", "Random Forest"),
  Training_Accuracy = c(0.8, 0.90, 0.90),  # Replace "NA" with actual training accuracy
  Validation_Accuracy = c(0.78, 0.69, 0.81),  # Replace "NA" with actual validation accuracy
  Overall_Accuracy = c(0.80, 0.86, 0.88)  # Replace "NA" with actual overall accuracy
)

# Print the table using kable
knitr::kable(accuracy_scores, caption = "Model Evaluation Results", align = "c")
```

Table 4: Model Evaluation Results

Model	Training_Accuracy	Validation_Accuracy	Overall_Accuracy
Decision Tree	0.8	0.78	0.80
Logistic Regression	0.9	0.69	0.86
Random Forest	0.9	0.81	0.88

It seems that compared to the logistic regression and decision tree models, the same variables shake out, predicting playoff participation (e.g., 3-point shooting percentage). Model performance is largely similar amongst the three approaches, with the random forrest approach showing the best predictive ability. However the easy to understand nature and visualization capabilities the decision tree make it an attractive alternative. While simple in nature, the

logistic regression model seems to perform well on the training data, however, does not do well with new, unseen data, suggesting overfitting